## AMENDMENTS TO THE CLAIMS

Claims 1-2, 4-22, 33-34, and 36-60 are currently pending. Claims 1, 9, 33, 41, and 55 are currently amended, without acquiescence in the cited basis for rejection or prejudice to pursue the original claims in a related application. A complete listing of the current pending claims is provided below and supersedes all previous claims listing(s). No new matter has been added.

1. (Currently Amended) A method for performing performance analysis for a target machine which comprise a software portion and a hardware portion, comprising:

describing a design for the target machine as a network of logical entities;

selecting at least one of the logical entities for a software implementation;

implementing a source software program for the logical entities selected for the software implementation;

generating an optimized assembler code for the software program, wherein the assembler code is an assembly-language representation of the software implementation;

performing a performance analysis using the assembler code;

generating a software simulation model based at least in part upon the assembler code by ~~translating the assembler code or~~ disassembling a binary code into a high level language format and by annotating the software simulation model with information related to ~~estimation or determination of the performance of~~ hardware on which the software program runs to capture a dynamic interaction between tasks during runtime, wherein the act of annotating the software simulation model is performed during a time when the act of generating a software simulation model by translating the assembler code or disassembling a binary code;

generating a hardware and software co-simulation model using the software simulation model; and

storing at least the hardware and software co-simulation model.

2. (Original) The method of claim 1, wherein the compiling step further comprises incorporating a description of the target machine.

3. (Cancelled).

4. (Previously Presented) The method of claim 1, further comprising selecting at least one of the logical entities for a hardware implementation, and using an existing software model of the hardware implementation from the selected logical entities, wherein the hardware and software co-simulation model is generated using the existing software model of the hardware implementation.

5. (Original) The method of claim 1, wherein the performance analysis measures an execution time of an element of the assembler code.

6. (Original) The method of claim 1, wherein the software program is compiled using the same compiler used to compile a production executable.

7. (Original) The method of claim 1, wherein performing the performance analysis comprises annotating the assembler code with performance information.

8. (Original) The method of claim 7, wherein the performance information is timing information.

9. (Currently Amended) A method of preparing software for a performance estimation, comprising:

obtaining a software assembly code module from a source code module, wherein the software assembly code module is an assembly-language representation;

generating a software simulation model by ~~translating the assembly code module~~ disassembling a binary code into the software simulation model in a high level language format;

annotating the software simulation model with performance information of a hardware together with which the software simulation model runs to capture a dynamic interaction between tasks during runtime, wherein the act of annotating the software simulation model is performed during a time when the act of

generating a software simulation model by disassembling a binary code into the software simulation model; and

storing at least the software simulation model,

wherein the software simulation model is an assembler-level software simulation model, expressed in a high-level programming language.

10. (Previously Presented) The method of claim 9, wherein providing a software assembly code module comprises compiling software source code to assembly.

11. (Previously Presented) The method of claim 10, wherein the software assembly code module is compiled using a compiler adapted to create code that will execute on a first machine architecture.

12. (Previously Presented) The method of claim 11, wherein the performance information is associated with the first machine architecture.

13. (Previously Presented) The method of claim 11, wherein the simulation model is compiled to execute on a second machine architecture, the second machine architecture being different from the first machine architecture.

14. (Previously Presented) The method of claim 1, wherein generating an optimized assembler code comprises disassembling software binary code to assembly code.

15. (Previously Presented) The method of claim 9, wherein the high-level programming language comprises a C code programming language.

16. (Previously Presented) The method of claim 9, wherein the translation step further comprises gathering information from the source code module from which the assembly code module was obtained.

17. (Previously Presented) The method of claim 16, wherein the information gathered comprises high-level hints about the software assembly code module.

18. (Previously Presented) The method of claim 9, wherein the performance information comprises estimated performance information.

19. (Previously Presented) The method of claim 9, wherein the performance information is statically estimated.

20. (Previously Presented) The method of claim 9, wherein the performance information is dynamically computed at run-time, using a formula provided during the annotating step.

21. (Previously Presented) The method of claim 9, further comprising:

compiling the simulation model to a simulator host program; and

executing the simulator host program on a simulator to allow performance measurements to be taken.

22. (Previously Presented) The method of claim 21, further comprising linking an already-annotated module with the simulation model.

23-32. (Cancelled).

33. (Currently Amended) A computer program product that includes a tangible volatile or non-volatile medium useable by a processor, the medium comprising a sequence of instructions which, when executed by said processor, causes said processor to execute a method for performing software performance analysis for a target machine, comprising:

describing a system design as a network of logical entities;

selecting at least one of the logical entities for a software implementation;

implementing a source software program for the logical entities selected for the software implementation;

compiling the software program to generate an optimized assembler code representation of the software program, wherein the optimized assembler code is an assembly-language representation of the software implementation;

performing a performance analysis using the assembler code;

generating a software simulation model based at least in part upon ~~by using~~ the assembler code by disassembling a binary code into a high level language format and by annotating the software simulation model with information related to hardware on which the software implementation runs ~~a result of the act of~~

~~performing performance analysis~~ to capture a dynamic interaction between tasks during runtime, wherein the act of annotating the software simulation model is performed during a time when the act of generating a software simulation model by disassembling the binary code;

generating a hardware and software co-simulation model using the software simulation model; and

storing at least the hardware and software co-simulation model.

34. (Previously Presented) The computer program product of claim 33, wherein the compiling step further comprises incorporating a description of the target machine.

35. (Cancelled).

36. (Previously Presented) The computer program product of claim 33, further comprising selecting at least one of the logical entities for a hardware implementation, and synthesizing a software model of the hardware implementation from the selected logical entities, wherein the hardware and software co-simulation model is generated using the software model of the hardware implementation.

37. (Previously Presented) The computer program product of claim 33, wherein the performance analysis measures an execution time of an element of the assembler code.

38. (Previously Presented) The computer program product of claim 33, wherein the software program is compiled using the same compiler used to compile a production executable.

39. (Previously Presented) The computer program product of claim 33, wherein performing the performance analysis comprises annotating the assembler code with performance information.

40. (Previously Presented) The computer program product of claim 39, wherein the performance information is timing information.

41. (Currently Amended) A computer program product that includes a tangible volatile or non-volatile medium useable by a processor, the medium comprising a sequence of

instructions which, when executed by said processor, causes said processor to execute a method for preparing software for a performance estimation, comprising:

obtaining a software assembly code module from a source code module, wherein the software assembly code module is an assembly-language representation;

generating a simulation model by ~~translating the assembly code module~~ disassembling a binary code into simulation model in a high level language format;

annotating the simulation model with performance information of hardware together with which the simulation model runs to capture a dynamic interaction between tasks during runtime, wherein the act of annotating the software simulation model is performed during a time when the act of generating a software simulation model by disassembling a binary code into the software simulation model; and

storing at least the simulation model,

wherein the simulation model is an assembler-level software simulation model, expressed in a high-level programming language.

42. (Previously Presented) The computer program product of claim 41, wherein providing a software assembly code module comprises compiling software source code to assembly.

43. (Previously Presented) The computer program product of claim 42, wherein the software assembly code module is compiled using a compiler adapted to create code that will execute on a first machine architecture.

44. (Previously Presented) The computer program product of claim 43, wherein the performance information is associated with the first machine architecture.

45. (Previously Presented) The computer program product of claim 43, wherein the simulation model is compiled to execute on a second machine architecture, the second machine architecture being different from the first machine architecture.

46. (Previously Presented) The computer program product of claim 41, wherein providing a software assembly code module comprises disassembling software binary code to assembly code.

47. (Previously Presented) The computer program product of claim 41, wherein the high-level programming language comprises a C code programming language.

48. (Previously Presented) The computer program product of claim 41, wherein the translation step further comprises gathering information from the source code module from which the assembly code module was obtained.

49. (Previously Presented) The computer program product of claim 48, wherein the information gathered comprises high-level hints about the software assembly code module.

50. (Previously Presented) The computer program product of claim 41, wherein the performance information comprises estimated performance information.

51. (Previously Presented) The computer program product of claim 41, wherein the performance information is statically estimated.

52. (Previously Presented) The computer program product of claim 41, wherein the performance information is dynamically computed at run-time, using a formula provided during the annotating step.

53. (Previously Presented) The computer program product of claim 41, further comprising:

compiling the simulation model to a simulator host program; and

executing the simulator host program on a simulator to allow performance measurements to be taken.

54. (Previously Presented) The computer program product of claim 53, further comprising linking an already-annotated module with the simulation model.

55. (Currently Amended) A method of translating an assembly language software module into a simulation model, comprising:

receiving the assembly language software module;

parsing the assembly language software module into a data structure, the data structure comprising one or more nodes, each of the one or more nodes being mapped to a period of time using a mapping definition, each of the one or more nodes containing an element of the assembly language software module;

processing the data structure to refine accuracy of an assembler-level software simulation model by generating the assembler-level software simulation model based on the assembly language software module by using the assembly language software module or by disassembling a binary code, wherein the assembler-level software simulation model is expressed in a high-level programming language and is used to determine a time slot;

associating performance information comprising a predicted execution delay with an element of the assembly language software module to capture a dynamic interaction between tasks during runtime, wherein the act of associating is performed during a time when the act of parsing the assembly language software into a data structure; and

displaying a result of the simulation model or storing the result of the simulation model in a tangible computer readable medium.

56. (Previously Presented) The method of claim 55, wherein the one or more nodes comprises a first node and a second node, the first node being mapped to a first period of time, the second node being mapped to a second period of time, the first period of time being different from the second period of time.

57. (Previously Presented) The method of claim 55, wherein the performance information comprises an execution delay value for the element of the assembly language software module.

58. (Previously Presented) The method of claim 55, wherein the performance information is a statically computed value.

59. (Previously Presented) The method of claim 55, wherein the performance information is a formula for dynamically computing a value.

60. (Previously Presented) The method of claim 55, wherein processing the data structure comprises replicating the behavior of the assembly language software model in the simulation model.